CMPSC 465 Data Structures & Algorithms
Spring 2024 Mehrdad Mahdavi and David Koslicki

HW 7

Due March 21, 10:00 pm

Instructions: You are encouraged to solve the problem sets on your own, or in groups of up to five people, but you must write your solutions strictly by yourself. You must explicitly acknowledge in your write-up all your collaborators, as well as any books, papers, web pages, etc. you got ideas from.

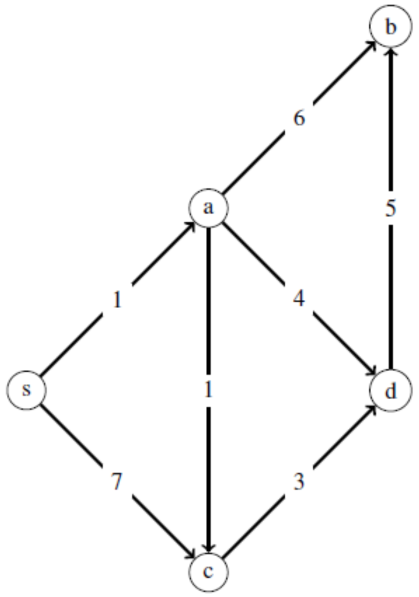
Formatting: Each problem should begin on a new page. Each page should be clearly labeled with the problem number. The pages of your homework submissions must be in order. You risk receiving no credit for it if you do not adhere to these guidelines.

Late homework will not be accepted. Please, do not ask for extensions since we will provide solutions shortly after the due date. Remember that we will drop your lowest two scores.

You need to submit it via Gradescope (Class code XX7RVV). Please ask on Campuswire about any details concerning Gradescope and formatting.

For each algorithm question, explain your algorithm and analyze its correctness and running time. Pseudocode is not required, but you may include it if you feel it makes your written explanation more clear.

1. (20 pts.) **Dijkstra's Algorithm.** Let $G = (V, E)$ be the following directed graph.



(a) Give the order of vertices in which they are removed from the min heap (priority queue) when Dijkstra's Algorithm is applied on G with s as the starting vertex.

(b) You are allowed to change the length of edge (a, d) but need to guarantee that, when Dijkstra's Algorithm is applied on the new graph (still with s as the starting vertex), the order of vertices being removed from the heap needs to be the same as before. Give the range (i.e., an interval) for a possible new length of edge (a, d) so that the above constraint can be satisfied.

2. (20 pts.) **Roads and planes.** In a country with n cities, we have two methods of travel: roads and planes. (More formally, our graph is $G = (V, R \cup P)$, where V is the set of cities, R is the set of roads, and P is the set of planes). Road $i \in R$ is specified as an **undirected** edge (u_i, v_i, C_i) that connects the two cities u_i, v_i with a **non-negative** cost C_i. Plane $j \in P$ is specified as **directed** edges (u_j, v_j) which connects city u_j to v_j (one way) with **non-negative** equal cost C due to some weird airline rewards program, with the additional constraint that there is no sequence of roads and planes that we can follow that takes us back from v_j to u_j. Also, for each maximal group of cities that can reach one another by only using roads, there is a major city among them that all planes arrive in due to security regulations, but the planes may leave from cities other than major cities. Also, in such a group, the shortest distance from the major city to any other city that a plane leaves from is always exactly C. Additionally, the total number of vertices that have at least 1 edge that is a road is $O(\sqrt{n})$.

Assume we have the adjacency list representation of roads and planes (two separate data structures), and we can freely swap between the two methods of travel (i.e. we can take an arbitrary amount of roads, then planes, then roads, and so on...).

Given a starting city s which is known to be able to reach any other city, give an efficient algorithm to compute the shortest path from s to every other city in linear time in the graph.

3. (20 pts.) **Start Node Negative Edges.** In general, Dijkstra's Algorithm fails on graphs with negative edges. However, consider running Dijkstra's Algorithm beginning at the only node with negative outgoing edges in the directed graph (all edges not leaving the start node are guaranteed to be positive). Assuming there are no negative cycles, either prove that Dijkstra's Algorithm functions correctly on all such graphs or provide an example graph where it fails.

4. (20 pts.) **Shortest Path in Currency Trading.** Shortest path algorithms can be applied in currency trading. Let $c_1, c_2, ..., c_n$ be various currencies; for instance, c_1 might be dollars, c_2 pounds, and c_3 lire. For any two currencies c_i and c_j, there is an exchange rate $r_{i,j}$; this means that you can purchase $r_{i,j}$ units of currency c_j in exchange for one unit of c_i. These exchange rates satisfy the condition that $r_{i,j} \cdot r_{j,i} < 1$, so that if you start with a unit of currency c_i, change it into currency c_j and then convert back to currency c_i, you end up with less than one unit of currency c_i (the difference is the cost of the transaction).

(a) Give an efficient algorithm for the following problem: Given a set of exchange rates $r_{i,j}$, and two currencies s and t, find the most advantageous sequence of currency exchanges for converting currency s into currency t. Toward this goal, you should represent the currencies and rates by a graph whose edge weights are real numbers.

(b) The exchange rates are updated frequently, reflecting the demand and supply of the various currencies. Occasionally the exchange rates satisfy the following property: there is a sequence of currencies $c_{i_1}, c_{i_2}, ..., c_{i_k}$ such that $r_{i_1,i_2} \cdot r_{i_2,i_3} \ldots r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1$. This means that by starting with a unit of currency c_{i_1} and then successively converting it to currencies $c_{i_2}, c_{i_3}, \ldots, c_{i_k}$, and finally back to c_{i_1}, you would end up with more than one unit of currency c_{i_1}. Such anomalies last only a fraction of a minute on the currency exchange, but they provide an opportunity for risk-free profits. Give an efficient algorithm for detecting the presence of such an anomaly. Use the graph representation you found above.

5. (20 pts.) **Faster All-Pairs Shortest Path.** The Floyd-Warshall Algorithm can find all pairs of shortest paths in $O(|V|^3)$ time. In this problem, we'll examine a different all-pairs shortest path algorithm based on running Dijkstra's Algorithm once from each node in the graph. For Dijkstra's Algorithm to function correctly, the graph must contain no negative edges. We accomplish this by reweighting every edge (u,v). Let $w(u,v)$ be the original weight and $\hat{w}(u,v) = w(u,v) + h(u) - h(v)$ be the new weight, where h is any function that maps the vertices to real numbers.

(a) Show that the shortest path $p = v_0, v_1, \cdots, v_{k-1}, v_k$ between vertices v_0 and v_k before the reweighting is still the shortest path from v_0 to v_k after the reweighting. (*Hint:* Consider the effect of the rewighting on the total weight of a path.)

(b) Show that for any cycle $c = v_0 v_1 \cdots v_{k-1} v_k$ where $v_k = v_0$, the total weight of the reweighted cycle is negative if and only if the total weight of the original cycle was also negative.

(c) Consider the following choice of h. Add a vertex s to the graph connected by zero-weight edges to every other vertex of the graph. Use Bellman-Ford algorithm to compute the shortest path from s to every other vertex and detect negative cycles. Assuming there are no negative cycles, let $h(v)$ be the shortest distance from s to v. Show that with this choice of h, every edge (u,v) in the graph has non-negative weight $\hat{w}(u,v) \geq 0$ after reweighting. (*Hint:* Triangle Inequality.)

(d) Following the reweighting procedure with the h described in part (c), the previous parts prove that we can run Dijkstra's Algorithm from every vertex to correctly compute all pairs of shortest paths. Assuming we use the $O(|V|\log|V| + |E|)$ Fibonacci-Heap implementation of Dijkstra's Algorithm, the total running time of this algorithm is $O(|V|^2 \log|V| + |V||E|)$, which is not always asymptotically faster than Floyd-Warshall. Explain what property the input graph must have for this algorithm to be asymptotically faster.

6. (0 pts.) **Acknowledgments.** The assignment will receive a 0 if this question is not answered.
(a) If you worked in a group, list the members of the group. Otherwise, write "I did not work in a group."
(b) If you received significant ideas about the HW solutions from anyone not in your group, list their names here. Otherwise, write "I did not consult with anyone other than my group members."
(c) List any resources besides the course material that you consulted in order to solve the material. If you did not consult anything, write "I did not consult any non-class materials."